

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification: G06F 17/21, G06F 3/00, G06F 13/00, G06F 17/30	A1	(11) International Publication Number: WO 00/04458 (43) International Publication Date: 27 January 2000 (27.01.2000)
---	-----------	--

(21) International Application Number:	PCT/US99/15951
(22) International Filing Date:	14 July 1999 (14.07.1999)
(30) Priority Data:	
09/314,863	19 May 1999 (19.05.1999) US
60/092,710	14 July 1998 (14.07.1998) US
(60) Parent Application or Grant	
MASSACHUSETTS INSTITUTE OF TECHNOLOGY [/]; O. LEIGHTON, F., Thomson [/]; O. LEWIN, Daniel, M. [/]; O . JUDSON, David, H. ; O.	

Published

- (54) Title: GLOBAL DOCUMENT HOSTING SYSTEM UTILIZING EMBEDDED CONTENT DISTRIBUTED GHOST SERVERS
(54) Titre: SYSTEME D'HEBERGEMENT GLOBAL DE DOCUMENTS FAISANT APPEL A UN CONTENU INTEGRE REPARTI
DANS DES SERVEURS FANTOMES

(57) Abstract

A network architecture or framework that supports hosting and content distribution on a truly global scale. The framework allows a Content Provider to replicate and serve its most popular content at an unlimited number of points throughout the world. The framework comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (36), sometimes referred to as ghost servers. This content comprises HTML page objects that, conventionally, are served from a Content Provider site. In accordance with the invention, however, a base HTML document portion of a web page is served from the Content Provider's site (1), while one or more embedded objects for the page are served from the hosting servers (3, 4), preferably those hosting servers (5) near the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

(57) Abrégé

L'invention concerne une architecture ou une ossature de réseau prenant en charge un hébergement et une répartition de contenu à une échelle réellement globale. L'ossature permet à un fournisseur de contenu de reproduire et de fournir son contenu le plus populaire à un nombre illimité de points à travers le monde. L'ossature comprend un ensemble de serveurs fonctionnant de manière répartie. Le contenu réel à fournir se trouve de préférence dans un ensemble de serveurs (36) hôtes, parfois dénommés serveurs fantômes. Ce contenu comprend des objets de page HTML qui, d'habitude sont fournis par un site de fournisseur de contenu. Dans le procédé selon l'invention, cependant, une partie de document HTML de base d'une page Web est fournie par le site (1) du fournisseur de contenu, alors qu'un ou plusieurs objets intégrés de la page sont fournis par les serveurs hôtes (3, 4), de préférence les serveurs hôtes (5) se trouvant à côté de la machine client. En fournissant le document HTML de base à partir du site de fournisseur de contenu, le fournisseur de contenu garde la maîtrise du contenu.

PCT

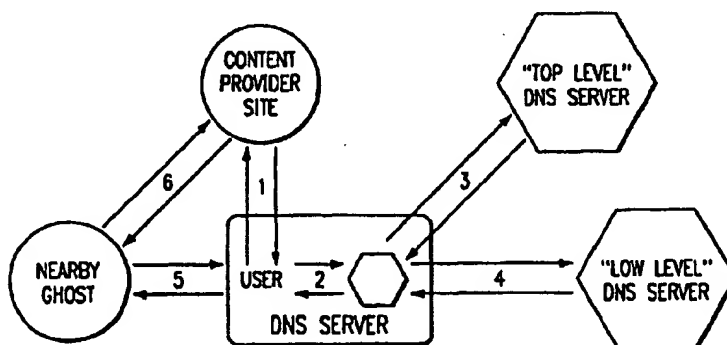
WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 17/21, 17/30, 3/00, 13/00		A1	(11) International Publication Number: WO 00/04458
			(43) International Publication Date: 27 January 2000 (27.01.00)
(21) International Application Number: PC1/US99/15951		(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).	
(22) International Filing Date: 14 July 1999 (14.07.99)			
(30) Priority Data: 60/092,710 14 July 1998 (14.07.98) US 09/314,863 19 May 1999 (19.05.99) US			
(71) Applicant: MASSACHUSETTS INSTITUTE OF TECHNOLOGY [US/US]; 77 Massachusetts Avenue N.E.-25-230, Cambridge, MA 02139 (US).		Published With international search report.	
(72) Inventors: LEIGHTON, F., Thomson; 15 Charlesden Park, Newtonville, MA 02160 (US). LEWIN, Daniel, M.; 292 Vassar Street #D6, Cambridge, MA 02139 (US).			
(74) Agent: JUDSON, David, H.; Hughes & Luce, L.L.P., 1717 Main Street, Suite 2800, Dallas, TX 75210 (US).			

(54) Title: GLOBAL DOCUMENT HOSTING SYSTEM UTILIZING EMBEDDED CONTENT DISTRIBUTED GHOST SERVERS



(57) Abstract

A network architecture or framework that supports hosting and content distribution on a truly global scale. The framework allows a Content Provider to replicate and serve its most popular content at an unlimited number of points throughout the world. The framework comprises a set of servers operating in a distributed manner. The actual content to be served is preferably supported on a set of hosting servers (36), sometimes referred to as ghost servers. This content comprises HTML page objects that, conventionally, are served from a Content Provider's site. In accordance with the invention, however, a base HTML document portion of a web page is served from the Content Provider's site (1), while one or more embedded objects for the page are served from the hosting servers (3, 4), preferably those hosting servers (5) near the client machine. By serving the base HTML document from the Content Provider's site, the Content Provider maintains control over the content.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AI.	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SW	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	ME	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

Description

5

10

15

20

25

30

35

40

45

50

55

**GLOBAL DOCUMENT HOSTING SYSTEM UTILIZING EMBEDDED CONTENT ..
DISTRIBUTED GHOST SERVERS**

Technical Field

This invention relates generally to information retrieval in a computer network. More particularly, the invention relates to a novel method of hosting and distributing content on the Internet that addresses the problems of Internet Service Providers (ISPs) and Internet Content Providers.

Description of the Related Art

The World Wide Web is the Internet's multimedia information retrieval system. In the Web environment, client machines effect transactions to Web servers using the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator or Microsoft Internet Explorer) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and, in return, receives a document or other object formatted according

5

to HTML. A collection of documents supported on a Web server is sometimes referred to as a Web site.

10

It is well known in the prior art for a Web site to mirror its content at another server. Indeed, at present, the only

5

method for a Content Provider to place its content closer to its readers is to build copies of its Web site on machines that are located at Web hosting farms in different locations domestically

15

and internationally. These copies of Web sites are known as

20

mirror sites. Unfortunately, mirror sites place unnecessary

10

economic and operational burdens on Content Providers, and they do not offer economies of scale. Economically, the overall cost

25

to a Content Provider with one primary site and one mirror site is more than twice the cost of a single primary site. This

30

additional cost is the result of two factors: (1) the Content

15

Provider must contract with a separate hosting facility for each mirror site, and (2) the Content Provider must incur additional

35

overhead expenses associated with keeping the mirror sites synchronized.

40

In an effort to address problems associated with mirroring, companies such as Cisco, Resonate, Bright Tiger, F5 Labs and Alteon, are developing software and hardware that will help keep

45

mirror sites synchronized and load balanced. Although these mechanisms are helpful to the Content Provider, they fail to address the underlying problem of scalability. Even if a

25

Content Provider is willing to incur the costs associated with

50

55

5

mirroring, the technology itself will not scale beyond a few
(i.e., less than 10) Web sites.

10

In addition to these economic and scalability issues, ^o
mirroring also entails operational difficulties. A Content

5 Provider that uses a mirror site must not only lease and manage

15

physical space in distant locations, but it must also buy and

maintain the software or hardware that synchronizes and load

balances the sites. Current solutions require Content Providers

20

to supply personnel, technology and other items necessary to

10 maintain multiple Web sites. In summary, mirroring requires

Content Providers to waste economic and other resources on

25

functions that are not relevant to their core business of

creating content.

30

Moreover, Content Providers also desire to retain control

15 of their content. Today, some ISPs are installing caching

hardware that interrupts the link between the Content Provider

35

and the end-user. The effect of such caching can produce

devastating results to the Content Provider, including (1)

preventing the Content Provider from obtaining accurate hit

40

20 counts on its Web pages (thereby decreasing revenue from

advertisers), (2) preventing the Content Provider from tailoring

content and advertising to specific audiences (which severely

45

limits the effectiveness of the Content Provider's Web page),

and (3) providing outdated information to its customers (which

25 can lead to a frustrated and angry end user).

50

55

5

10

15

There remains a significant need in the art to provide a decentralized hosting solution that enables users to obtain Internet content on a more efficient basis (i.e., without burdening network resources unnecessarily) and that likewise enables the Content Provider to maintain control over its content.

20

The present invention solves these and other problems associated with the prior art.

BRIEF SUMMARY OF THE INVENTION

25

30

10 It is a general object of the present invention to provide a computer network comprising a large number of widely deployed Internet servers that form an organic, massively fault-tolerant infrastructure designed to serve Web content efficiently, effectively, and reliably to end users.

35

40

15 Another more general object of the present invention is to provide a fundamentally new and better method to distribute Web-based content. The inventive architecture provides a method for intelligently routing and replicating content over a large network of distributed servers, preferably with no centralized control.

45

50

Another object of the present invention is to provide a network architecture that moves content close to the user. The inventive architecture allows Web sites to develop large audiences without worrying about building a massive infrastructure to handle the associated traffic.

55

5

10

15

Still another object of the present invention is to provide a fault-tolerant network for distributing Web content. The network architecture is used to speed-up the delivery of richer Web pages, and it allows Content Providers with large audiences to serve them reliably and economically, preferably from servers located close to end users.

20

10

A further feature of the present invention is the ability to distribute and manage content over a large network without disrupting the Content Provider's direct relationship with the end user.

25

30

15

Yet another feature of the present invention is to provide a distributed scalable infrastructure for the Internet that shifts the burden of Web content distribution from the Content Provider to a network of preferably hundreds of hosting servers deployed, for example, on a global basis.

35

40

20

In general, the present invention is a network architecture that supports hosting on a truly global scale. The inventive framework allows a Content Provider to replicate its most popular content at an unlimited number of points throughout the world. As an additional feature, the actual content that is replicated at any one geographic location is specifically tailored to viewers in that location. Moreover, content is automatically sent to the location where it is requested, without any effort or overhead on the part of a Content Provider.

45

50

25

55

5

It is thus a more general object of this invention to ..
provide a global hosting framework to enable Content Providers
to retain control of their content.

10

The hosting framework of the present invention comprises a
5 set of servers operating in a distributed manner. The actual
15 content to be served is preferably supported on a set of hosting
servers (sometimes referred to as ghost servers). This content
comprises HTML page objects that, conventionally, are served
20 from a Content Provider site. In accordance with the invention,
10 however, a base HTML document portion of a Web page is served
from the Content Provider's site while one or more embedded
25 objects for the page are served from the hosting servers,
preferably, those hosting servers nearest the client machine.
By serving the base HTML document from the Content Provider's
30 site, the Content Provider maintains control over the content.
15

30

The determination of which hosting server to use to serve a
35 given embedded object is effected by other resources in the
hosting framework. In particular, the framework includes a
second set of servers (or server resources) that are configured
40 to provide top level Domain Name Service (DNS). In addition,
the framework also includes a third set of servers (or server
resources) that are configured to provide low level DNS
45 functionality. When a client machine issues an HTTP request to
the Web site for a given Web page, the base HTML document is
25 served from the Web site as previously noted. Embedded objects

50

55

5 for the page preferably are served from particular hosting
servers identified by the top- and low-level DNS servers. To
10 locate the appropriate hosting servers to use, the top-level DNS
server determines the user's location in the network to identify
5 a given low-level DNS server to respond to the request for the
15 embedded object. The top-level DNS server then redirects the
request to the identified low-level DNS server that, in turn,
resolves the request into an IP address for the given hosting
20 server that serves the object back to the client.

10 More generally, it is possible (and, in some cases,
desirable) to have a hierarchy of DNS servers that consisting of
25 several levels. The lower one moves in the hierarchy, the closer
one gets to the best region.

30 A further aspect of the invention is a means by which
15 content can be distributed and replicated through a collection
of servers so that the use of memory is optimized subject to the
constraints that there are a sufficient number of copies of any
35 object to satisfy the demand, the copies of objects are spread
so that no server becomes overloaded, copies tend to be located
40 on the same servers as time moves forward, and copies are
located in regions close to the clients that are requesting
them. Thus, servers operating within the framework do not keep
45 copies of all of the content database. Rather, given servers
keep copies of a minimal amount of data so that the entire
25 system provides the required level of service. This aspect of

5

the invention allows the hosting scheme to be far more efficient than schemes that cache everything everywhere, or that cache objects only in prespecified locations.

10

The global hosting framework is fault tolerant at each level of operation. In particular, the top level DNS server returns a list of low-level DNS servers that may be used by the client to service the request for the embedded object. Likewise, each hosting server preferably includes a buddy server that is used to assume the hosting responsibilities of its associated hosting server in the event of a failure condition.

15

20

10

25

According to the present invention, load balancing across the set of hosting servers is achieved in part through a novel technique for distributing the embedded object requests. In particular, each embedded object URL is preferably modified by prepending a virtual server hostname into the URL. More generally, the virtual server hostname is inserted into the URL. Preferably, the virtual server hostname includes a value (sometimes referred to as a serial number) generated by applying a given hash function to the URL or by encoding given information about the object into the value. This function serves to randomly distribute the embedded objects over a given set of virtual server hostnames. In addition, a given fingerprint value for the embedded object is generated by applying a given hash function to the embedded object itself. This given value serves as a fingerprint that identifies whether

30

35

40

45

50

55

5 the embedded object has been modified. Preferably, the
functions used to generate the values (i.e., for the virtual
10 server hostname and the fingerprint) are applied to a given Web
page in an off-line process. Thus, when an HTTP request for the
5 page is received, the base HTML document is served by the Web
15 site and some portion of the page's embedded objects are served
from the hosting servers near (although not necessarily the
closest) to the client machine that initiated the request.

20 The foregoing has outlined some of the more pertinent
10 objects and features of the present invention. These objects
should be construed to be merely illustrative of some of the
25 more prominent features and applications of the invention. Many
other beneficial results can be attained by applying the
disclosed invention in a different manner or modifying the
30 invention as will be described. Accordingly, other objects and
a fuller understanding of the invention may be had by referring
35 to the following Detailed Description of the Preferred
Embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

40 20 For a more complete understanding of the present invention
and the advantages thereof, reference should be made to the
following Detailed Description taken in connection with the
45 accompanying drawings in which:

Figure 1 is a representative system in which the present
25 invention is implemented;

5

10

Figure 2 is a simplified representation of a markup language document illustrating the base document and a set of embedded objects;

Figure 3 is a high level diagram of a global hosting system according to the present invention;

15

Figure 4 is a simplified flowchart illustrating a method of processing a Web page to modified embedded object URLs that is used in the present invention;

20

Figure 5 is a simplified state diagram illustrating how the present invention responds to a HTTP request for a Web page.

25

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

30

35

40

45

50

55

A known Internet client-server system is implemented as illustrated in **Figure 1**. A client machine 10 is connected to a Web server 12 via a network 14. For illustrative purposes, network 14 is the Internet, an intranet, an extranet or any other known network. Web server 12 is one of a plurality of servers which are accessible by clients, one of which is illustrated by machine 10. A representative client machine includes a browser 16, which is a known software tool used to access the servers of the network. The Web server supports files (collectively referred to as a "Web" site) in the form of hypertext documents and objects. In the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL).

5 A representative Web server 12 is a computer comprising a
processor 18, an operating system 20, and a Web server program
10 22, such as Netscape Enterprise Server. The server 12 also
includes a display supporting a graphical user interface (GUI)
5 for management and administration, and an Application
15 Programming Interface (API) that provides extensions to enable
application developers to extend and/or customize the core
functionality thereof through software programs including Common
20 Gateway Interface (CGI) programs, plug-ins, servlets, active
10 server pages, server side include (SSI) functions or the like.

25 A representative Web client is a personal computer that is
x86-, PowerPC®- or RISC-based, that includes an operating system
such as IBM® OS/2® or Microsoft Windows '95, and that includes a
30 Web browser, such as Netscape Navigator 4.0 (or higher), having
15 a Java Virtual Machine (JVM) and support for application plug-
ins or helper applications. A client may also be a notebook
35 computer, a handheld computing device (e.g., a PDA), an Internet
appliance, or any other such device connectable to the computer
network.

40 20 As seen in Figure 2, a typical Web page comprises a markup
language (e.g. HTML) master or base document 28, and many
embedded objects (e.g., images, audio, video, or the like) 30.
45 Thus, in a typical page, twenty or more embedded images or
objects are quite common. Each of these images is an
25 independent object in the Web, retrieved (or validated for
50
55

5 change) separately. The common behavior of a Web client, .
therefore, is to fetch the base HTML document, and then
10 immediately fetch the embedded objects, which are typically (but
not always) located on the same server. According to the
5 present invention, preferably the markup language base document
15 28 is served from the Web server (i.e., the Content Provider
site) whereas a given number (or perhaps all) of the embedded
objects are served from other servers. As will be seen,
20 preferably a given embedded object is served from a server
10 (other than the Web server itself) that is close to the client
machine, that is not overloaded, and that is most likely to
25 already have a current version of the required file.

Referring now to **Figure 3**, this operation is achieved by
the hosting system of the present invention. As will be seen,
30 15 the hosting system 35 comprises a set of widely-deployed servers
(or server resources) that form a large, fault-tolerant
35 infrastructure designed to serve Web content efficiently,
effectively, and reliably to end users. The servers may be
deployed globally, or across any desired geographic regions. As
40 20 will be seen, the hosting system provides a distributed
architecture for intelligently routing and replicating such
content. To this end, the global hosting system 35 comprises
45 three (3) basic types of servers (or server resources): hosting
servers (sometimes called ghosts) 36, top-level DNS servers 38,
25 and low-level DNS servers 40. Although not illustrated, there
50

5 may be additional levels in the DNS hierarchy. Alternatively,
there may be a single DNS level that combines the functionality
10 of the top level and low-level servers. In this illustrative
embodiment, the inventive framework 35 is deployed by an
5 Internet Service Provider (ISP), although this is not a
limitation of the present invention. The ISP or ISPs that
15 deploy the inventive global hosting framework 35 preferably have
a large number of machines that run both the ghost server
20 component 36 and the low-level DNS component 40 on their
10 networks. These machines are distributed throughout the
network; preferably, they are concentrated around network
25 exchange points 42 and network access points 44, although this
is not a requirement. In addition, the ISP preferably has a
small number of machines running the top-level DNS 38 that may
30 also be distributed throughout the network.

Although not meant to be limiting, preferably a given
35 server used in the framework 35 includes a processor, an
operating system (e.g., Linux, UNIX, Windows NT, or the like), a
Web server application, and a set of application routines used
40 by the invention. These routines are conveniently implemented
in software as a set of instructions executed by the processor
to perform various process or method steps as will be described
45 in more detail below. The servers are preferably located at the
edges of the network (e.g., in points of presence, or POPs).

5

10

15

Several factors may determine where the hosting servers are placed in the network. Thus, for example, the server locations are preferably determined by a demand driven network map that allows the provider (e.g., the ISP) to monitor traffic requests. By studying traffic patterns, the ISP may optimize the server locations for the given traffic profiles.

20

25

30

35

According to the present invention, a given Web page (comprising a base HTML document and a set of embedded objects) is served in a distributed manner. Thus, preferably, the base HTML document is served from the Content Provider that normally hosts the page. The embedded objects, or some subset thereof, are preferentially served from the hosting servers 36 and, specifically, given hosting servers 36 that are near the client machine that in the first instance initiated the request for the Web page. In addition, preferably loads across the hosting servers are balanced to ensure that a given embedded object may be efficiently served from a given hosting server near the client when such client requires that object to complete the page.

40

45

50

To serve the page contents in this manner, the URL associated with an embedded object is modified. As is well-known, each embedded object that may be served in a page has its own URL. Typically, the URL has a hostname identifying the Content Provider's site from where the object is conventionally served, i.e., without reference to the present invention.

55

5

10

According to the invention, the embedded object URL is first modified, preferably in an off-line process, to condition the URL to be served by the global hosting servers. A flowchart illustrating the preferred method for modifying the object URL is illustrated in **Figure 4**.

15

20

25

30

35

40

45

50

55

The routine begins at step 50 by determining whether all of the embedded objects in a given page have been processed. If so, the routine ends. If not, however, the routine gets the next embedded object at step 52. At step 54, a virtual server hostname is prepended into the URL for the given embedded object. The virtual server hostname includes a value (e.g., a number) that is generated, for example, by applying a given hash function to the URL. As is well-known, a hash function takes arbitrary length bit strings as inputs and produces fixed length bit strings (hash values) as outputs. Such functions satisfy two conditions: (1) it is infeasible to find two different inputs that produce the same hash value, and (2) given an input and its hash value, it is infeasible to find a different input with the same hash value. In step 54, the URL for the embedded object is hashed into a value xx,xxx that is then included in the virtual server hostname. This step randomly distributes the object to a given virtual server hostname.

The present invention is not limited to generating the virtual server hostname by applying a hash function as described above. As an alternative and preferred embodiment, a virtual

5 server hostname is generated as follows. Consider the
representative hostname al234.g.akamaitech.net. The 1234 value,
10 sometimes referred to as a serial number, preferably includes
information about the object such as its size (big or small),
5 its anticipated popularity, the date on which the object was
15 created, the identity of the Web site, the type of object (e.g.,
movie or static picture), and perhaps some random bits generated
by a given random function. Of course, it is not required that
20 any given serial number encode all of such information or even a
10 significant number of such components. Indeed, in the simplest
case, the serial number may be a simple integer. In any event,
25 the information is encoded into a serial number in any
convenient manner. Thus, for example, a first bit is used to
30 denote size, a second bit is used to denote popularity, a set of
15 additional bits is used to denote the date, and so forth. As
noted above in the hashing example, the serial number is also
35 used for load balancing and for directing certain types of
traffic to certain types of servers. Typically, most URLs on
the same page have the same serial number to minimize the number
40 of distinguished name (DN) accesses needed per page. This
requirement is less important for larger objects.

Thus, according to the present invention, a virtual server
45 hostname is prepended into the URL for a given embedded object,
and this hostname includes a value (or serial number) that is
25 generated by applying a given function to the URL or object.

5

That function may be a hash function, an encoding function, or the like.

10

Turning now back to the flowchart, the routine then continues at step 56 to include a given value in the object's URL. Preferably, the given value is generated by applying a given hash function to the embedded object. This step creates a unique fingerprint of the object that is useful for determining whether the object has been modified. Thereafter, the routine returns to step 50 and cycles.

15

20

With the above as background, the inventive global hosting framework is now described in the context of a specific example. In particular, it is assumed that a user of a client machine in Boston requests a Content Provider Web page normally hosted in Atlanta. For illustrative purposes, It is assumed that the Content Provider is using the global hosting architecture within a network, which may be global, international, national, regional, local or private. **Figure 5** shows the various components of the system and how the request from the client is processed. This operation is not to be taken by way of limitation, as will be explained.

25

30

35

40

45

Step 1: The browser sends a request to the Provider's Web site (Item 1). The Content Provider site in Atlanta receives the request in the same way that it does as if the global hosting framework were not being implemented. The difference is in what is returned by the Provider site. Instead of returning

50

55

5

10

the usual page, according to the invention, the Web site returns a page with embedded object URLs that are modified according to the method illustrated in the flowchart of **Figure 4**. As previously described, the URLs preferably are changed as

5 follows:

15

Assume that there are 100,000 virtual ghost servers, even though there may only be a relatively small number (e.g., 100) physically present on the network. These virtual ghost servers or virtual ghosts are identified by the hostname:

20

10 ghostxxxxx.ghosting.com, where xxxxx is replaced by a number between 0 and 99,999. After the Content Provider Web site is updated with new information, a script executing on the Content Provider site is run that rewrites the embedded URLs.

25

30

15 Preferably, the embedded URLs names are hashed into numbers between 0 and 99,999, although this range is not a limitation of the present invention. An embedded URL is then switched to reference the virtual ghost with that number. For example, the following is an embedded URL from the Provider's site:

35

40

20 If the serial number for the object referred to by this URL is the number 1467, then preferably the URL is rewritten to read:

45

<IMG SRC = http:

//ghost1467.ghosting.akamai.com/www.provider.com/TECH/images/space.story.gif>.

25

50

55

5

10

15

The use of serial numbers in this manner distributes the embedded URLs roughly evenly over the 100,000 virtual ghost server names. Note that the Provider site can still personalize the page by rearranging the various objects on the screen according to individual preferences. Moreover, the Provider can also insert advertisements dynamically and count how many people view each ad.

20

25

30

According to the preferred embodiment, an additional modification to the embedded URLs is made to ensure that the global hosting system does not serve stale information. As previously described, preferably a hash of the data contained in the embedded URL is also inserted into the embedded URL itself. That is, each embedded URL may contain a fingerprint of the data to which it points. When the underlying information changes, so does the fingerprint, and this prevents users from referencing old data.

35

40

45

50

The second hash takes as input a stream of bits and outputs what is sometimes referred to as a fingerprint of the stream. The important property of the fingerprint is that two different streams almost surely produce two different fingerprints. Examples of such hashes are the MD2 and MD5 hash functions, however, other more transparent methods such as a simple checksum may be used. For concreteness, assume that the output of the hash is a 128 bit signature. This signature can be interpreted as a number and then inserted into the embedded URL.

55

For example, if the hash of the data in the picture
space.story.gif from the Provider web site is the number 28765,
then the modified embedded URL would actually look as follows:

<IMG

SRC=http://ghost1467.ghosting.akamai.com/28765/www.provider.com
/TECH/images/space.story.gif">.

Whenever a page is changed, preferably the hash for each
embedded URL is recomputed and the URL is rewritten if
necessary. If any of the URL's data changes, for example, a new
and different picture is inserted with the name space.story.gif,
then the hash of the data is different and therefore the URL
itself will be different. This scheme prevents the system from
serving data that is stale as a result of updates to the
original page.

For example, assume that the picture space.story.gif is
replaced with a more up-to-date version on the Content Provider
server. Because the data of the pictures changes, the hash of
the URL changes as well. Thus, the new embedded URL looks the
same except that a new number is inserted for the fingerprint.
Any user that requests the page after the update receives a page
that points to the new picture. The old picture is never
referenced and cannot be mistakenly returned in place of the
more up-to-date information.

In summary, preferably there are two hashing operations
that are done to modify the pages of the Content Provider.

5 First, hashing can be a component of the process by which a
serial number is selected to transform the domain name into a
10 virtual ghost name. As will be seen, this first transformation
serves to redirect clients to the global hosting system to
5 retrieve the embedded URLs. Next, a hash of the data pointed to
15 by the embedded URLs is computed and inserted into the URL.
This second transformation serves to protect against serving
stale and out-of-date content from the ghost servers.
20 Preferably, these two transformations are performed off-line and
10 therefore do not pose potential performance bottlenecks.

Generalizing, the preferred URL schema is as follows. The
25 illustrative domain `www.domainname.com/frontpage.jpg` is
transformed into:

30 `xxxx.yy.zzzz.net/aaaa/www.domainname.com/frontpage.jpg,`

15 where:

`xxxx` = serial number field

35 `yy` = lower level DNS field

`zzzz` = top level DNS field

`aaaa` = other information (e.g., fingerprint) field.

40 20 If additional levels of the DNS hierarchy are used, then
there may be additional lower level DNS fields, e.g.,
`xxxx.y1y1.y2y2.zzz.net/aaaa/. . .`

45 Step 2: After receiving the initial page from the Content
Provider site, the browser needs to load the embedded URLs to
25 display the page. The first step in doing this is to contact

50

55

5

the DNS server on the user's machine (or at the user's ISP) to resolve the altered hostname, in this case:

10

ghost1467.ghosting.akamai.com. As will be seen, the global hosting architecture of the present invention manipulates the DNS system so that the name is resolved to one of the ghosts that is near the client and is likely to have the page already. To appreciate how this is done, the following describes the progress of the DNS query that was initiated by the client.

15

20

Step 3: As previously described, preferably there are two types of DNS servers in the inventive system: top-level and low-level. The top level DNS servers 38 for ghosting.com have a special function that is different from regular DNS servers like those of the .com domain. The top level DNS servers 38 include appropriate control routines that are used to determine where in the network a user is located, and then to direct the user to a akamai.com (i.e., a low level DNS) server 40 that is close-by. Like the .com domain, akamai.com preferably has a number of top-level DNS servers 38 spread throughout the network for fault tolerance. Thus, a given top level DNS server 38 directs the user to a region in the Internet (having a collection of hosting servers 36 that may be used to satisfy the request for a given embedded object) whereas the low level DNS server 40 (within the identified region) identifies a particular hosting server within that collection from which the object is actually served.

25

30

15

35

40

45

50

55

5

10

15

20

25

30

35

40

45

50

55

More generally, as noted above, the DNS process can contain several levels of processing, each of which serves to better direct the client to a ghost server. The ghost server name can also have more fields. For example, "a123.g.g.akamaitech.net" may be used instead of "a123.ghost.akamai.com." If only one DNS level is used, a representative URL could be "a123.akamai.com."

Although other techniques may be used, the user's location in the network preferably is deduced by looking at the IP address of the client machine making the request. In the present example, the DNS server is running on the machine of the user, although this is not a requirement. If the user is using an ISP DNS server, for example, the routines make the assumption that the user is located near (in the Internet sense) this server. Alternatively, the user's location or IP address could be directly encoded into the request sent to the top level DNS. To determine the physical location of an IP address in the network, preferably, the top level DNS server builds a network map that is then used to identify the relevant location.

Thus, for example, when a request comes in to a top level DNS for a resolution for a1234.g.akamaitech.net, the top level DNS looks at the return address of the requester and then formulates the response based on that address according to a network map. In this example, the a1234 is a serial number, the g is a field that refers to the lower level DNS, and akamaitech refers to the top level DNS. The network map preferably

5

contains a list of all Internet Protocol (IP) blocks and, for each IP block, the map determines where to direct the request.

10

The map preferably is updated continually based on network conditions and traffic.

5

15

After determining where in the network the request originated, the top level DNS server redirects the DNS request to a low level DNS server close to the user in the network. The ability to redirect requests is a standard feature in the DNS system. In addition, this redirection can be done in such a way that if the local low level DNS server is down, there is a backup server that is contacted.

25

30

15

Preferably, the TTL (time to live) stamp on these top level DNS redirections for the ghosting.com domain is set to be long. This allows DNS caching at the user's DNS servers and/or the ISP's DNS servers to prevent the top level DNS servers from being overloaded. If the TTL for ghosting.akamai.com in the DNS server at the user's machine or ISP has expired, then a top level server is contacted, and a new redirection to a local low level ghosting.akamai.com DNS server is returned with a new TTL stamp. It should be noted the system does not cause a substantially larger number of top level DNS lookups than what is done in the current centralized hosting solutions. This is because the TTL of the top level redirections are set to be high and, thus, the vast majority of users are directed by their

40

45

50

55

5

local DNS straight to a nearby low level ghosting.akamai.com DNS server.

10

Moreover, fault tolerance for the top level DNS servers⁶ is provided automatically by DNS similarly to what is done for the popular .com domain. Fault tolerance for the low level DNS servers preferably is provided by returning a list of possible low level DNS servers instead of just a single server. If one of the low level DNS servers is down, the user will still be able to contact one on the list that is up and running.

15

20

10

Fault tolerance can also be handled via an "overflow control" mechanism wherein the client is redirected to a low-level DNS in a region that is known to have sufficient capacity to serve the object. This alternate approach is very useful in scenarios where there is a large amount of demand from a specific region or when there is reduced capacity in a region. In general, the clients are directed to regions in a way that minimizes the overall latency experienced by clients subject to the constraint that no region becomes overloaded. Minimizing overall latency subject to the regional capacity constraints preferably is achieved using a min-cost multicommodity flow algorithm.

25

30

15

35

40

20

45

Step 4: At this point, the user has the address of a close-by ghosting.com DNS server 38. The user's local DNS server contacts the close-by low level DNS server 40 and requests a translation for the name ghost1467.ghosting.akamai.com. The

50

55

5

10

local DNS server is responsible for returning the IP address of one of the ghost servers 36 on the network that is close to the user, not overloaded, and most likely to already have the required data.

15

20

25

30

5 The basic mechanism for mapping the virtual ghost names to real ghosts is hashing. One preferred technique is so-called consistent hashing, as described in U.S. Serial No. 09/042,228, filed March 13, 1998, and in U.S. Serial No. 09/088,825, filed June 2, 1998, each titled Method And Apparatus For Distributing Requests Among A Plurality Of Resources, and owned by the Massachusetts Institute of Technology, which applications are incorporated herein by reference. Consistent hash functions make the system robust under machine failures and crashes. It also allows the system to grow gracefully, without changing where most items are located and without perfect information about the system.

35

40

45

50

According to the invention, the virtual ghost names may be hashed into real ghost addresses using a table lookup, where the table is continually updated based on network conditions and traffic in such a way to insure load balancing and fault tolerance. Preferably, a table of resolutions is created for each serial number. For example, serial number 1 resolves to ghost 2 and 5, serial number 2 resolves to ghost 3, serial number 3 resolves to ghosts 2,3,4, and so forth. The goal is to define the resolutions so that no ghost exceeds its capacity and

55

5

10

15

20

that the total number of all ghosts in all resolutions is minimized. This is done to assure that the system can take maximal advantage of the available memory at each region. This is a major advantage over existing load balancing schemes that tend to cache everything everywhere or that only cache certain objects in certain locations no matter what the loads are. In general, it is desirable to make assignments so that resolutions tend to stay consistent over time provided that the loads do not change too much in a short period of time. This mechanism preferably also takes into account how close the ghost is to the user, and how heavily loaded the ghost is at the moment.

25

30

35

Note that the same virtual ghost preferably is translated to different real ghost addresses according to where the user is located in the network. For example, assume that ghost server 18.98.0.17 is located in the United States and that ghost server 132.68.1.28 is located in Israel. A DNS request for ghost1487.ghosting.akamai.com originating in Boston will resolve to 18.98.0.17, while a request originating in Tel-Aviv will resolve to 132.68.1.28.

40

45

The low-level DNS servers monitor the various ghost servers to take into account their loads while translating virtual ghost names into real addresses. This is handled by a software routine that runs on the ghosts and on the low level DNS servers. In one embodiment, the load information is circulated among the servers in a region so that they can compute

50

55

5 resolutions for each serial number. One algorithm for computing
resolutions works as follows. The server first computes the
10 projected load (based on number of user requests) for each
serial number. The serial numbers are then processed in
5 increasing order of load. For each serial number, a random
15 priority list of desired servers is assigned using a consistent
hashing method. Each serial number is then resolved to the
smallest initial segment of servers from the priority list so
20 that no server becomes overloaded. For example, if the priority
10 list for a serial number is 2,5,3,1,6, then an attempt is made
first to try to map the load for the serial number to ghost 2.
25 If this overloads ghost 2, then the load is assigned to both
ghosts 2 and 5. If this produced too much load on either of
those servers, then the load is assigned to ghosts 2,3, and 5,
30 and so forth. The projected load on a server can be computed by
looking at all resolutions that contain that server and by
35 adding the amount of load that is likely to be sent to that
server from that serial number. This method of producing
resolutions is most effective when used in an iterative fashion,
40 wherein the assignments starts in a default state, where every
serial number is mapped to every ghost. By refining the
resolution table according to the previous procedure, the load
45 is balanced using the minimum amount of replication (thereby
maximally conserving the available memory in a region).

5

10

15

20

25

30

35

40

45

50

55

The TTL for these low level DNS translations is set to be short to allow a quick response when heavy load is detected on one of the ghosts. The TTL is a parameter that can be manipulated by the system to insure a balance between timely response to high load on ghosts and the load induced on the low level DNS servers. Note, however, that even if the TTL for the low level DNS translation is set to 1-2 minutes, only a few of the users actually have to do a low level DNS lookup. Most users will see a DNS translation that is cached on their machine or at their ISP. Thus, most users go directly from their local DNS server to the close-by ghost that has the data they want. Those users that actually do a low level DNS lookup have a very small added latency, however this latency is small compared to the advantage of retrieving most of the data from close by.

As noted above, fault tolerance for the low level DNS servers is provided by having the top level DNS return a list of possible low level DNS servers instead of a single server address. The user's DNS system caches this list (part of the standard DNS system), and contacts one of the other servers on the list if the first one is down for some reason. The low level DNS servers make use of a standard feature of DNS to provide an extra level of fault tolerance for the ghost servers. When a name is translated, instead of returning a single name, a list of names is returned. If for some reason the primary fault tolerance method for the ghosts (known as the Buddy system,

5 which is described below) fails, the client browser will contact
one of the other ghosts on the list.

10 Step 5: The browser then makes a request for an object
named a123.ghosting.akamai.com/.../www.provider.com/TECH/
5 images/space.story.gif from the close-by ghost. Note that the
15 name of the original server (www.provider.com) preferably is
included as part of the URL. The software running on the ghost
parses the page name into the original host name and the real
20 page name. If a copy of the file is already stored on the
ghost, then the data is returned immediately. If, however, no
25 copy of the data on the ghost exists, a copy is retrieved from
the original server or another ghost server. Note that the
ghost knows who the original server was because the name was
30 encoded into the URL that was passed to the ghost from the
15 browser. Once a copy has been retrieved it is returned to the
user, and preferably it is also stored on the ghost for
35 answering future requests.

As an additional safeguard, it may be preferable to check
that the user is indeed close to the server. This can be done
40 20 by examining the IP address of the client before responding to
the request for the file. This is useful in the rare case when
the client's DNS server is far away from the client. In such a
45 case, the ghost server can redirect the user to a closer server
(or to another virtual address that is likely to be resolved to
25 a server that is closer to the client). If the redirect is to a
50

5

10

virtual server, then it must be tagged to prevent further
redirections from taking place. In the preferred embodiment,
redirection would only be done for large objects; thus, a check
may be made before applying a redirection to be sure that the
object being requested exceeds a certain overall size.

15

20

25

30

35

40

45

50

55

Performance for long downloads can also be improved by
dynamically changing the server to which a client is connected
based on changing network conditions. This is especially
helpful for audio and video downloads (where the connections can
be long and where quality is especially important). In such
cases, the user can be directed to an alternate server in mid-
stream. The control structure for redirecting the client can be
similar to that described above, but it can also include
software that is placed in the client's browser or media player.
The software monitors the performance of the client's connection
and perhaps the status of the network as well. If it is deemed
that the client's connection can be improved by changing the
server, then the system directs the client to a new server for
the rest of the connection.

Fault tolerance for the ghosts is provided by a buddy
system, where each ghost has a designated buddy ghost. If a
ghost goes down, its buddy takes over its work (and IP address)
so that service is not interrupted. Another feature of the
system is that the buddy ghost does not have to sit idle waiting
for a failure. Instead, all of the machines are always active,

5

10

and when a failure happens, the load is taken over by the buddy and then balanced by the low level DNS system to the other active ghosts. An additional feature of the buddy system is that fault tolerance is provided without having to wait for long timeout periods.

15

20

25

30

35

As yet another safety feature of the global hosting system, a gating mechanism can be used to keep the overall traffic for certain objects within specified limits. One embodiment of the gating mechanism works as follows. When the number of requests for an object exceeds a certain specified threshold, then the server can elect to not serve the object. This can be very useful if the object is very large. Instead, the client can be served a much smaller object that asks the client to return later. Or, the client can be redirected. Another method of implementing a gate is to provide the client with a "ticket" that allows the client to receive the object at a prespecified future time. In this method, the ghost server needs to check the time on the ticket before serving the object.

40

45

50

The inventive global hosting scheme is a way for global ISPs or conglomerates of regional ISPs to leverage their network infrastructure to generate hosting revenue, and to save on network bandwidth. An ISP offering the inventive global hosting scheme can give content providers the ability to distribute content to their users from the closest point on the ISPs network, thus ensuring fast and reliable access. Guaranteed web

55

5

site performance is critical for any web-based business, and
global hosting allows for the creation of a service that
satisfies this need.

10

Global hosting according to the present invention also
allows an ISP to control how and where content traverses its
network. Global hosting servers can be set up at the edges of
the ISP's network (at the many network exchange and access
points, for example). This enables the ISP to serve content for
sites that it hosts directly into the network exchange points
and access points. Expensive backbone links no longer have to
carry redundant traffic from the content provider's site to the
network exchange and access points. Instead, the content is
served directly out of the ISP's network, freeing valuable
network resources for other traffic.

20

25

30

Although global hosting reduces network traffic, it is also
a method by which global ISPs may capture a piece of the rapidly
expanding hosting market, which is currently estimated at over a
billion dollars a year.

35

The global hosting solution also provides numerous
advantages to Content Providers, and, in particular, an
efficient and cost-effective solution to improve the performance
of their Web sites both domestically and internationally. The
inventive hosting software ensures Content Providers with fast
and reliable Internet access by providing a means to distribute
content to their subscribers from the closest point on an ISP's

40

45

50

55

5

network. In addition to other benefits described in more detail below, the global hosting solution also provides the important benefit of reducing network traffic.

10

15

20

25

30

Once inexpensive global hosting servers are installed at the periphery of an ISP's network (i.e., at the many network exchange and access points), content is served directly into network exchange and access points. As a result of this efficient distribution of content directly from an ISP's network, the present invention substantially improves Web site performance. In contrast to current content distribution systems, the inventive global hosting solution does not require expensive backbone links to carry redundant traffic from the Content Provider's Web site to the network exchange and access points.

A summary of the specific advantages afforded by the inventive global hosting scheme are set forth below:

35

1. Decreased Operational Expenses for Content Providers:

40

45

Most competing solutions require Content Providers to purchase servers at each Web site that hosts their content. As a result, Content Providers often must negotiate separate contracts with different ISPs around the world. In addition, Content Providers are generally responsible for replicating the content and maintaining servers in these remote locations.

50

With the present invention, ISPs are primarily responsible for the majority of the aspects of the global hosting. Content

55

5 Providers preferably maintain only their single source server.
Content on this server is automatically replicated by software
10 to the locations where it is being accessed. No intervention or
planning is needed by the Provider (or, for that matter, the
5 ISP). Content Providers are offered instant access to all of
15 the servers on the global network; there is no need to choose
where content should be replicated or to purchase additional
servers in remote locations.

20 2. Intelligent and Efficient Data Replication:

10 Most competing solutions require Content Providers to
replicate their content on servers at a commercial hosting site
25 or to mirror their content on geographically distant servers.
Neither approach is particularly efficient. In the former
situation, content is still located at a single location on the
30 Internet (and thus it is far away from most users). In the
latter case, the entire content of a Web site is copied to
35 remote servers, even though only a small portion of the content
may actually need to be located remotely. Even with inexpensive
memory, the excessive cost associated with such mirroring makes
40 20 it uneconomical to mirror to more than a few sites, which means
that most users will still be far away from a mirror site.
Mirroring also has the added disadvantage that Content Providers
45 must insure that all sites remain consistent and current, which
is a nontrivial task for even a few sites.

5

10

15

With the present invention, content is automatically replicated to the global server network in an intelligent and efficient fashion. Content is replicated in only those locations where it is needed. Moreover, when the content changes, new copies preferably are replicated automatically throughout the network.

3. Automatic Content Management:

20

25

30

35

Many existing solutions require active management of content distribution, content replication and load balancing between different servers. In particular, decisions about where content will be hosted must be made manually, and the process of replicating data is handled in a centralized push fashion. On the contrary, the invention features passive management. Replication is done in a demand-based pull fashion so that content preferably is only sent to where it is truly needed. Moreover, the process preferably is fully automated; the ISP does not have to worry about how and where content is replicated and/or the content provider.

4. Unlimited, Cost Effective Scalability:

40

45

50

55

Competing solutions are not scalable to more than a small number of sites. For example, solutions based on mirroring are typically used in connection with at most three or four sites. The barriers to scaling include the expense of replicating the entire site, the cost of replicating computing resources at all

5

nodes, and the complexity of supporting the widely varying ..
software packages that Content Providers use on their servers.

10

The unique system architecture of the present invention is
scaleable to hundreds, thousands or even millions of nodes.

5

15

Servers in the hosting network can malfunction or crash and the
system's overall function is not affected. The global hosting
framework makes efficient use of resources; servers and client
software do not need to be replicated at every node because only
the hosting server runs at each node. In addition, the global
hosting server is designed to run on standard simple hardware
that is not required to be highly fault tolerant.

25

5. Protection against Flash Crowds:

30

15

35

40

20

Competing solutions do not provide the Content Provider
with protection from unexpected flash crowds. Although mirroring
and related load-balancing solutions do allow a Content Provider
to distribute load across a collection of servers, the aggregate
capacity of the servers must be sufficient to handle peak
demands. This means that the Provider must purchase and
maintain a level of resources commensurate with the anticipated
peak load instead of the true average load. Given the highly
variable and unpredictable nature of the Internet, such
solutions are expensive and highly wasteful of resources.

45

25

The inventive hosting architecture allows ISPs to utilize a
single network of hosting servers to offer Content Providers
flash crowd insurance. That is, insurance that the network will

50

55

5 automatically adapt to and support unexpected higher load on the
Provider's site. Because the ISP is aggregating many Providers
10 together on the same global network, resources are more
efficiently used.

5 6. Substantial Bandwidth Savings:

15 Competing solutions do not afford substantial bandwidth
savings to ISPs or Content Providers. Through the use of
mirroring, it is possible to save bandwidth over certain links
20 (i.e., between New York and Los Angeles). Without global
10 hosting, however, most requests for content will still need to
transit the Internet, thus incurring bandwidth costs. The
25 inventive hosting framework saves substantial backbone bandwidth
for ISPs that have their own backbones. Because content is
distributed throughout the network and can be placed next to
30 network exchange points, both ISPs and Content Providers
15 experience substantial savings because backbone charges are not
incurred for most content requests.

35 7. Instant Access to the Global Network:

40 Competing solutions require the Content Provider to choose
20 manually a small collection of sites at which content will be
hosted and/or replicated. Even if the ISP has numerous hosting
sites in widely varied locations, only those sites specifically
45 chosen (and paid for) will be used to host content for that
Content Provider.

5 On the contrary, the global hosting solution of the present
invention allows ISPs to offer their clients instant access to
10 the global network of servers. To provide instant access to the
global network, content is preferably constantly and dynamically
5 moved around the network. For example, if a Content Provider
15 adds content that will be of interest to customers located in
Asia, the Content Provider will be assured that its content will
be automatically moved to servers that are also located in Asia.
20 In addition, the global hosting framework allows the content to
10 be moved very close to end users (even as close as the user's
building in the case of the Enterprise market).

25 8. Designed for Global ISPs and Conglomerates:

Most competing solutions are designed to be purchased and
30 managed by Content Providers, many of whom are already
15 consistently challenged and consumed by the administrative and
operational tasks of managing a single server. The inventive
35 hosting scheme may be deployed by a global ISP, and it provides
a new service that can be offered to Content Providers. A
feature of the service is that it minimizes the operational and
40 20 managerial requirements of a Content Provider, thus allowing the
Content Provider to focus on its core business of creating
unique content.

45 9. Effective Control of Proprietary Databases and
Confidential Information:

5

10

15

20

25

Many competing solutions require Content Providers to ..
replicate their proprietary databases to multiple geographically
distant sites. As a result, the Content Provider effectively
loses control over its proprietary and usually confidential
databases. To remedy these problems, the global hosting
solution of the present invention ensures that Content Providers
retain complete control over their databases. As described
above, initial requests for content are directed to the Content
Provider's central Web site, which then implements effective and
controlled database access. Preferably, high-bandwidth, static
parts for page requests are retrieved from the global hosting
network.

10. Compatibility with Content Provider Software:

30

35

40

45

50

Many competing solutions require Content Providers to
utilize a specific set of servers and databases. These
particular, non-uniform requirements constrain the Content
Provider's ability to most effectively use new technologies, and
may require expensive changes to a Content Provider's existing
infrastructure. By eliminating these problems, the inventive
global hosting architecture effectively interfaces between the
Content Provider and the ISP, and it does not make any
assumptions about the systems or servers used by the Content
Provider. Furthermore, the Content Provider's systems can be
upgraded, changed or completely replaced without modifying or
interrupting the inventive architecture.

55

5

11. No Interference with Dynamic Content, Personalized Advertising or E-Commerce, and No stale content:

10

15

20

25

30

Many competing solutions (such as naive caching of all content) can interfere with dynamic content, personalized advertising and E-commerce and can serve the user with stale content. While other software companies have attempted to partially eliminate these issues (such as keeping counts on hits for all cached copies), each of these solutions causes a partial or complete loss of functionality (such as the ability to personalize advertising). On the contrary, the global hosting solution does not interfere with generation of dynamic content, personalized advertising or E-commerce, because each of these tasks preferably is handled by the central server of the Content Provider.

12. Designed for the Global Network:

35

The global hosting architecture is highly scaleable and thus may be deployed on a world-wide network basis.

40

45

50

The above-described functionality of each of the components of the global hosting architecture preferably is implemented in software executable in a processor, namely, as a set of instructions or program code in a code module resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a

55

5

CD ROM) or floppy disk (for eventual use in a floppy disk ..
drive), or downloaded via the Internet or other computer
network.

10

15

20

In addition, although the various methods described are
5 conveniently implemented in a general purpose computer
selectively activated or reconfigured by software, one of
ordinary skill in the art would also recognize that such methods
may be carried out in hardware, in firmware, or in more
specialized apparatus constructed to perform the required method
10 steps.

25

30

35

40

45

50

55

Further, as used herein, a Web "client" should be broadly
construed to mean any computer or component thereof directly or
indirectly connected or connectable in any known or later-
developed manner to a computer network, such as the Internet.
15 The term Web "server" should also be broadly construed to mean a
computer, computer platform, an adjunct to a computer or
platform, or any component thereof. Of course, a "client"
should be broadly construed to mean one who requests or gets the
file, and "server" is the entity which downloads the file.

20 Having thus described our invention, what we claim as new
and desire to secure by Letters Patent is set forth in the
following claims.

Claims

5

10

15

20

25

30

35

40

45

50

55

5

CLAIMS

10

1. A method of serving a Web page comprising a markup language base document and a set of embedded objects each of which is identified by a URL, wherein given embedded objects each include a URL that has been modified to include a virtual server hostname, the method comprising the steps of:

15

responsive to a request for the Web page issued from a client machine, serving the base document to the client machine from a content provider site; and

20

serving the given embedded objects to the client machine from hosting servers identified by the virtual server hostnames.

25

30

2. The method as described in Claim 1 wherein the hosting servers are located in a computer network near the client machine.

15

35

3. The method as described in Claim 1 wherein the step of serving a given embedded object to the client machine further includes the steps of:

40

redirecting the request from a first level domain name server to a second level domain name server near the client machine; and

45

having the second level domain name server resolve the virtual server hostname to identify a given set of one or more hosting servers for serving the embedded object.

50

55

5

10

4. The method as described in Claim 1 wherein the virtual server hostname includes a value generated by applying a given function to the embedded object.

5

15

5. The method as described in Claim 4 wherein the value is generated by encoding given information, the given information selected from a group of information consisting essentially of: size data, popularity data, creation data and object type data.

20

10

25

6. The method as described in Claim 1 wherein the modified URL includes a fingerprint value generated by applying a given function to the embedded object.

30

15

35

7. The method as described in Claim 6 wherein the value is a number generated by hashing the embedded object.

40

8. The method as described in Claim 1 wherein the markup language is HTML.

45

9. The method as described in Claim 1 further including the step of rewriting the modified URLs as the content provider modifies the Web page.

25

50

55

5

10

15

20

25

30

35

40

45

50

55

10. A method of serving a Web page comprising a markup language base document and a set of embedded objects, each embedded object identified by a URL, comprising the steps of: prepending a hostname into the URL for a given embedded object, server hostname including a value generated by applying a given function to the embedded object; and in response to a request from a client browser, serving the web page.

11. The method as described in Claim 10 wherein the hostname value is generated by encoding given information, the given information selected from a group of information consisting essentially of: size data, popularity data, creation data and object type data.

12. The method as described in Claim 10 further including the step of: modifying the URL to include a fingerprint value for the embedded object by applying a given function to the embedded object.

13. The method as described in Claim 12 wherein the given fingerprint value is generated by hashing the embedded object.

5

10

14. A method of processing a Web page comprising a . . .
hypertext markup language base document and a set of embedded
objects, each embedded object identified by a URL, comprising
the steps of:

15

5 prepending a virtual server hostname into the URL for a
given embedded object, the virtual server hostname including a
value generated by applying a given function to the URL or the
given object;

20

wherein the given function randomly distributes the
10 embedded objects over a given set of virtual server hostnames.

25

15 15. The method as described in Claim 14 wherein the given
function is an encoding function.

30

16. The method as described in Claim 14 wherein the given
function is a hash function.

35

17. The method as described in Claim 14 further including
the step of:

40

including in the URL a given fingerprint value for the
20 embedded object generated by applying a given hash function to
the embedded object;

45

wherein the given fingerprint value identifies whether the
embedded object has been modified.

50

55

5

10

18. The method as described in Claim 17 further including the step of recomputing the hash for the fingerprint value and rewriting the URL if necessary when the Web page is changed.

15

20

5 19. A distributed hosting framework operative in a computer network in which users of client machines connect to a server via service providers, wherein the server supports pages each comprising a markup language base document and a set of embedded objects and wherein each embedded object is identified by a URL, the framework comprising:

25

a first set of servers that host the embedded objects;

at least one top level server that provides a top level domain name service (DNS) resolution; and

30

at least one lower level server that provides a lower level domain name service (DNS) resolution;

35

wherein page requests generated by the client machines are serviced by the server and a given subset of the first set of servers as identified by the top level and lower level servers.

40

20 20. The hosting framework as described in Claim 19 further including a redundant top level server.

45

21. The hosting framework as described in Claim 19 further including a redundant lower level server.

25

50

55

5

10

5

15

20

25

30

35

40

45

50

55

22. The hosting framework as described in Claim 19 wherein a given one of the first set of servers includes a buddy server for assuming the hosting responsibilities of the given one of the first set of servers upon a given failure condition.

23. The hosting framework as described in Claim 19 wherein the lower level server includes a load balancing mechanism that balances loads across a subset of the first set of servers.

24. The hosting framework as described in Claim 23 wherein the load balancing mechanism minimizes the amount of replication required for the embedded objects while not exceeding a capacity of any of the first set of servers.

25. The hosting framework as described in Claim 19 further including an overflow control mechanism for minimizing an overall amount of latency experienced by client machines while not exceeding the capacity of any given subset of the first set of servers.

26. The hosting framework as described in Claim 25 wherein the overflow control mechanism includes a min-cost multicommodity flow algorithm.

5

10

27. The hosting framework as described in Claim 19 wherein the top level server includes a network map for use in directing a page request generated by a client to a given one of the first set of servers.

5

15

20

28. The hosting framework as described in Claim 19 wherein a server in the first set of server includes a gating mechanism for maintaining overall traffic for a given embedded object within specified limits.

10

25

29. The hosting framework as described in Claim 28 wherein the gating mechanism comprises:

30

means for determining whether a number of requests for the given embedded object exceeds a given threshold; and

15

means responsive to the determining means for restricting service of the given embedded object.

35

40

20

30. The hosting framework as described in Claim 29 wherein the restricting means comprises means for serving an object that is smaller than the given embedded object.

45

31. The hosting framework as described in Claim 29 wherein the object is a ticket that allows a client to receive the given embedded object at a later time.

50

55

5

10

15

20

25

30

32. A method of serving a Web page comprising a markup language base document and a set of embedded objects each of which is identified by a URL, wherein given embedded objects each include a URL that has been modified to include a virtual server hostname, the method comprising the steps of:

responsive to requests for the Web page issued from first and second client machines, serving the base document to each client machine from a content provider site;

serving a given embedded object to the first client machine from a first hosting server identified by a first virtual server hostname; and

serving the given embedded object to the second client machine from a second hosting server also identified by the first virtual server hostname.

15

35

40

33. The method as described in Claim 32 further including the step of resolving the first virtual server hostname into an address for the first hosting server as a function of a location of the first client machine and local traffic conditions.

20

45

50

34. The method as described in Claim 32 further including the steps of:

as the given embedded object is being served to the first client machine, determining whether the given embedded object can be served more efficiently from another hosting server; and

55

5 if so, serving a remainder of the given embedded object to
the first client machine from a third hosting server.

10
35. A method of serving a Web page comprising a markup
5 language base document and a set of embedded objects each of
15 which is identified by a URL, wherein given embedded objects
each include a URL that has been modified to include a virtual
server hostname, the method comprising the steps of:
20 responsive to requests for the Web page issued from first
10 and second client machines, serving the base document to each
client machine from a content provider site;
25 resolving the virtual server hostname into a first address
and a second address;
30 serving a given embedded object to the first client machine
15 from a first hosting server located at the first address; and
serving the given embedded object to the second client
35 machine from a second hosting server located at the second
address.

40 20 36. The method as described in Claim 35 wherein the
virtual server hostname is resolved to the first address or the
second address according to where the request for the Web page
45 originates and local traffic conditions in an associated region
of the computer network.

25

50

55

5

10

15

20

10

25

30

35

40

45

50

55

37. A method of serving a Web page comprising a markup language base document and a set of embedded objects, each embedded object identified by a URL, comprising the steps of:

rewriting the URL of an embedded object to generate a modified URL, the modified URL including a new hostname prepended to an original hostname, wherein the original hostname is maintained as part of the modified URL for use in retrieving the embedded object whenever a cached copy of the embedded object is not available; and

in response to a request to serve the Web page received from a client browser, serving the Web page.

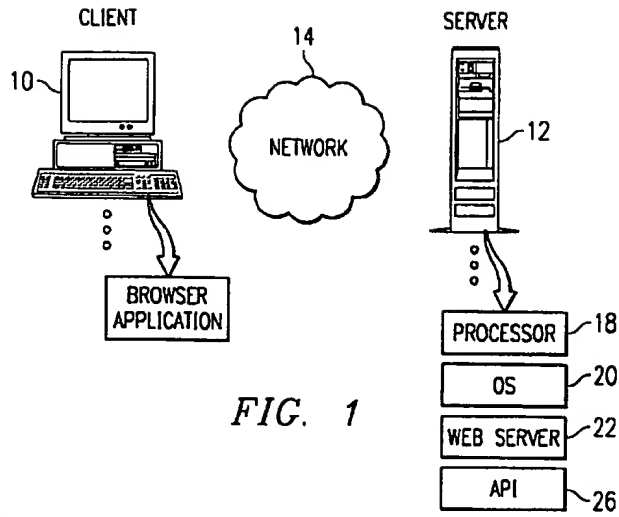


FIG. 1

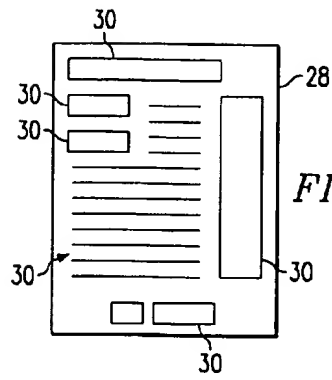


FIG. 2

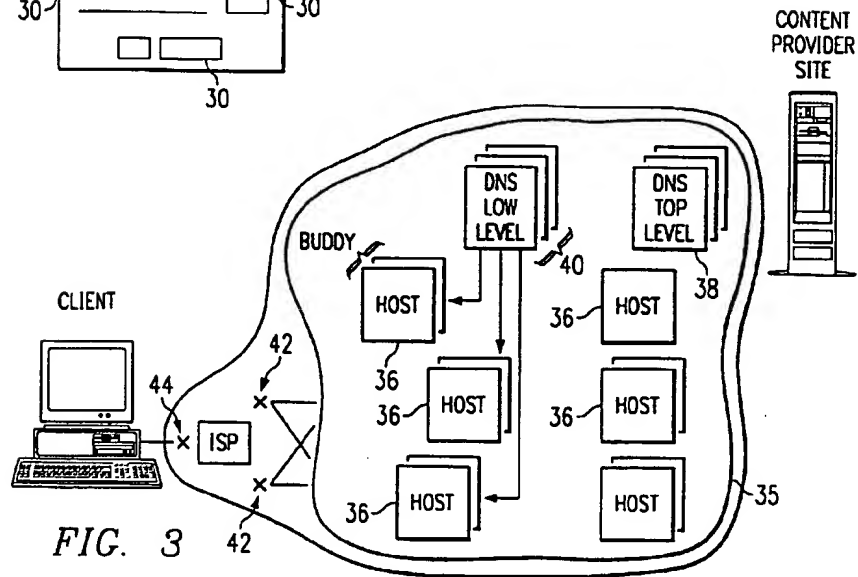


FIG. 3

FIG. 4

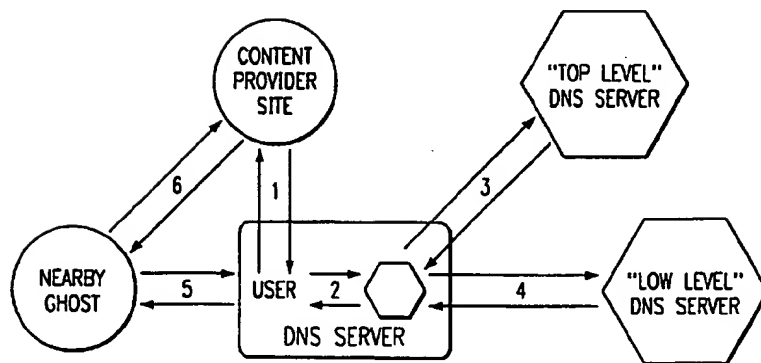
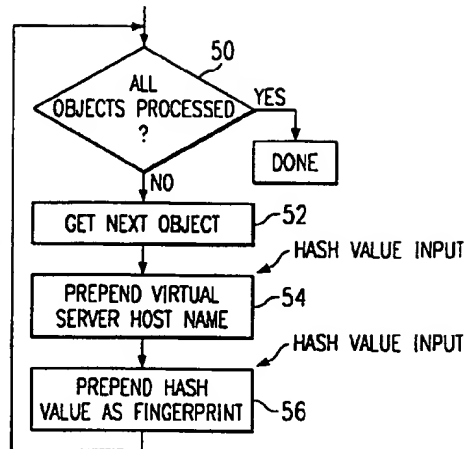


FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/15951

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 17/21, 17/30, 3/00, 13/00 US CL : 707/501, 10, 203; 709/201, 219; 345/329 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 707/501, 10, 203; 709/201, 219; 345/329 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) APS, DIALOG, IS&R search terms: URL, HTTP, HTML, modify, replace, ghost, dynamic, mirror		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,E	US 5,945,989 A (FREISHTAT et al) 31 August 1999	1-37
A,E	US 5,933,832 A (SUZUOKA et al) 03 August 1999	1-37
A,P	US 5,903,723 A (BECK et al) 11 May 1999	1-37
A,P	US 5,870,559 A (LESHEM et al) 09 February 1999	1-37
A,P	US 5,832,506 A (KUZMA) 03 November 1998	1-37
A	US 5,751,961 A (SMYK) 12 May 1998	1-37
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "B" earlier document published on or after the international filing date "L" document which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "A" document member of the same patent family		
Date of the actual completion of the international search 29 SEPTEMBER 1999		Date of mailing of the international search report 21 OCT 1999
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer MICHAEL RAZAVI <i>James R. Matthews</i> Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet)(July 1992)*